STRUCTure: Towards a Distributed Boolean Satisfiability Solver

Alexandru Mosoi
ami650@few.vu.nl

July 25, 2011

Supervisor: Jason Maassen jason@cs.vu.nl
Supervisor: Kees Verstoep c.verstoep@vu.nl

Layout

Problem Definition

A **Boolean formula**, $F : \mathbb{B}^n \rightarrow \mathbb{B}$, is in Conjuctive Normal Form if it is a conjuction of disjunctions of literals.

- Variable: $1 \ldots N$
  - Can take one of two truth values $True$ or $False$
- Literal: a or $\neg a$, a variable
  - $1, \neg 3$
- Clause: $u_1 \vee u_2 \ldots \vee u_k$, u_i literals
  - $1 \vee 2 \vee \neg 4 \vee \neg 5 \vee 7$
- Formula: $C_1 \wedge C_2 \ldots \wedge C_m$, C_i clauses
  - $(\mathbf{1} \vee \neg \mathbf{2}) \wedge (\mathbf{1} \vee \mathbf{3}) \wedge (\mathbf{2} \vee \mathbf{3} \vee \mathbf{5})$
- Any formula can be transformed such that clauses have at most 3 literals ($3 - SAT$ which is in NP).

Goal

Goal

Find a satifying assignment for any Boolean formula in CNF, or return that formula is inconsistent

Example

- $(1 \vee \neg 2) \wedge (1 \vee 3) \wedge (2 \vee 3 \vee 5)$
- $1 = \text{False}$, $2 = \text{False}$, $3 = \text{True}$, $5 = \text{False}$
- $\neg 1, \neg 2, 3, \neg 5$

Importance

Academia

- One of Karp's 21 **NP-complete** problem [1]
- Other NP-complete problems can be reduced to $3 - SAT$
  - e.g: graph coloring, vertex cover

Industry

- Circuit encoding of Boolean logic [2]
  - planning, synthesis, biology, verification, testing, routing, ...

---

[1] Richard M. Karp (1972). *Reducibility Among Combinatorial Problems.* http://www.cs.berkeley.edu/~luca/cs172/karp.pdf

[2] G. Audemard, L. Sas. *Circuit Based Encoding of CNF formula*

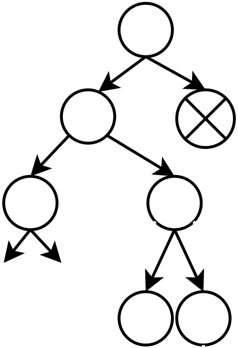
Parallelization

- State-of-art solvers are *sequential*
  - but they are wicked fast
- Current approaches to parallelism:
  - Run same solvers with different settings
    - Pingeling, SAT4J [3]
  - Share database of learned clauses (usually limited to units and, maybe, binaries)
    - Pingeling [4] and Cryptominisat [5]
  - Don't scale well
- We want: *Horizontal scalability* [6]
  - Replicated (not shared) memory
  - Less synchronization

[3] SAT4J. `http://www.sat4j.org/`
[4] Pingeling. `http://fmv.jku.at/papers/Biere-FMV-TR-10-1.pdf`
[5] Cryptominisat. `http://www.msoos.org/cryptominisat2`
[6] *Horizontal scalability* is Scalability in number of cpus

Layout

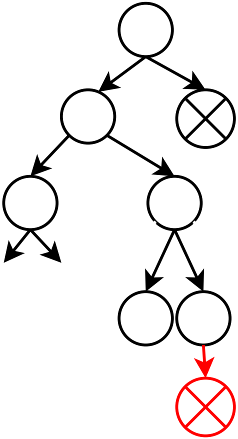
Types of solvers



- Backtracking (DPLL)
  - Pick a variable, u
  - For both polarities $u, \neg u$:
    - Simplify the formula
    - If f. is empty return solution (empty formula is consistent)
    - else if f. is inconsistent backtrack (found a *conflict*)
    - else recurse
- Conflict Driven/Clause Learning
- Look-ahead

Davis, Martin; Logemann, George, and Loveland, Donald (1962). *A Machine Program for Theorem Proving*

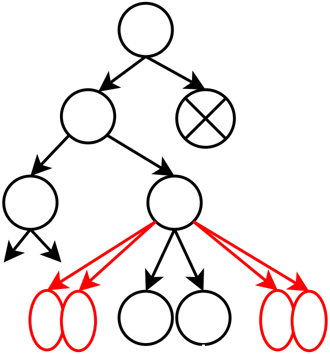
Types of solvers



- Backtracking (DPLL)
- Conflict Driven/Clause Learning
  - Searches conflicts
  - *Learns* new clauses to *reduce* search space
  - Maintain a database of learned clauses
- Look-ahead

Davis, Martin; Logemann, George, and Loveland, Donald (1962). *A Machine Program for Theorem Proving*

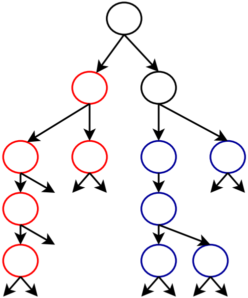
Types of solvers



- Backtracking (DPLL)
- Conflict Driven/Clause Learning
- Look-ahead
  - Picks variables which *simplify* the formula as much as possible
  - Eventually the formula is *empty* (consistent) or contains an *empty clause* (inconsistent)
- STRUCTure is Look-ahead + Clause Learning

Davis, Martin; Logemann, George, and Loveland, Donald (1962). *A Machine Program for Theorem Proving*

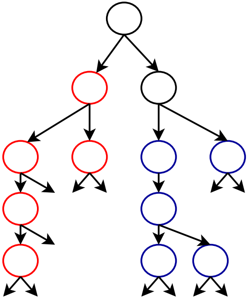
Constellation



- Computational model based on activities which:
  - perform work
  - spawn new activities
  - send/process events to/from other activities

J. Maassen, F.J. Seinstra, H.E. Bal, *Context Aware Many-Task Computing with Ibis/Constellation*

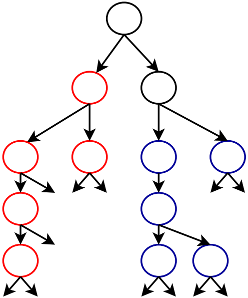
Constellation



- Each node (machine) has a fixed number of executors (threads) which run activities.
- *Work stealing*
  - Local: *small jobs*
  - Remote: *large jobs*

J. Maassen, F.J. Seinstra, H.E. Bal, *Context Aware Many-Task Computing with Ibis/Constellation*

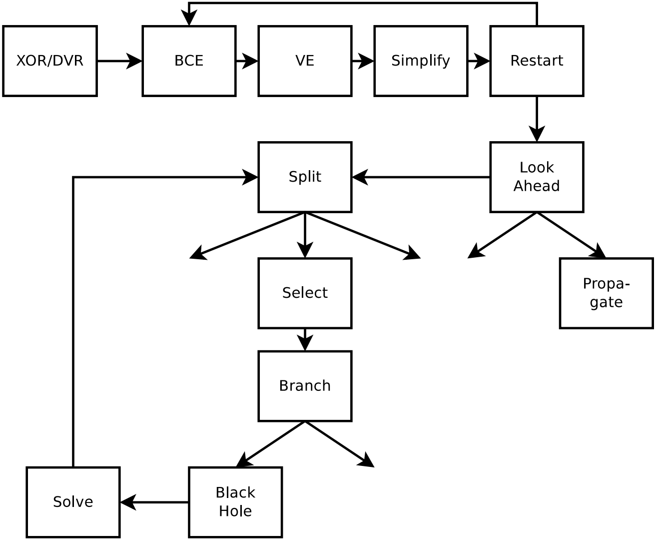
Constellation



- Advantage:
  - Easy to parallelize (no synchronization)
  - Scalable
- Disadvantage:
  - No *shared memory* or *broadcasting*
  - *No cancel*

J. Maassen, F.J. Seinstra, H.E. Bal, *Context Aware Many-Task Computing with Ibis/Constellation*

Architecture of STRUCTure



XOR Gates and Dependent Variable Removal [9] [10]

- Many instances encode several logical gates: *OR, AND, XOR*.
- *XOR* with N inputs: $\mathbf{2^N}$ CNF clauses of $N+1$ literals

Idea

Decode XOR gates from formula and use them as eqns. in $\mathbb{Z}_2$.

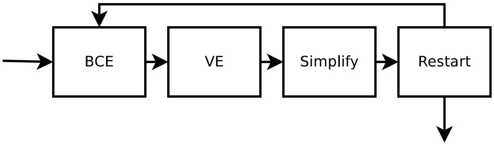
Dependent Variable Removal

- Variable is dependent if it appears *only* in XOR gates.
- Can be removed together with some clauses.

[9] M.J.H. Heule. *Towards a lookahead sat solver for general purposes.*

[10] J.A. Roy, I.L. Markov. *Restoring circuit structure from sat instances.*

Restart Loop



Idea

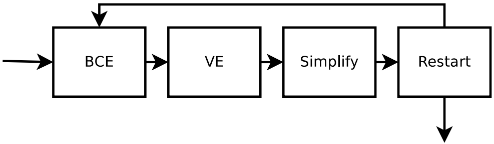
Restart computation, maybe a different variable selection order leads faster to a solution.

How

- Simplifies the formula
  - Blocked Clause Elimination [11]
  - Variable Elimination
  - Simplify

[11] M Jarvisalo, A. Biere, M. Heule, *Blocked Clause Elimination*

Restart Loop



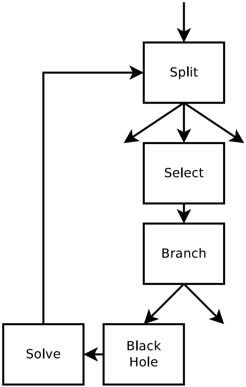
Idea

Restart computation, maybe a different variable selection order leads faster to a solution.

How

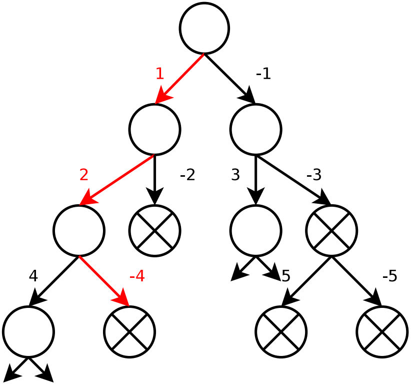
- Simplifies the formula
- Starts and stops searching after some time
- Extend formula with learned clauses

Searching



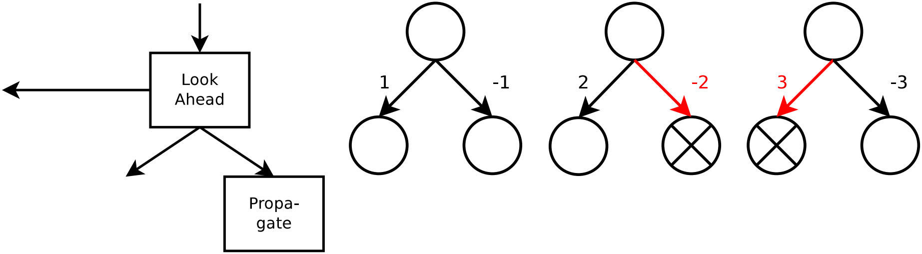
- *Split* divides formula into a disjunction of independent smaller formulas
- *Select* picks a variable for branching
  - analysis of clauses in which the variables appear
- *Branch* is the backtracking step
- *BlackHole* filters out instances from old restarts
- *Solve* simplifies the formula with newest added branch

Learning



- *Conflict*: cannot assign selected literals $1, 2, \neg 4$
- $\overline{(1 \wedge 2 \wedge \neg 4)} \equiv (\neg 1 \vee \neg 2 \vee 4)$
- $(\neg 1 \vee 2), (1 \vee 3)$
- Next time no need to search the same space again.
- Assign $\neg 1$ then 3 must be false

Look-ahead



Idea

- Perform search from root until depth 1
- *Learn units* and sometimes binaries
  - unit = clause of length 1 = variable assignment
- Then solve problem with newest clauses

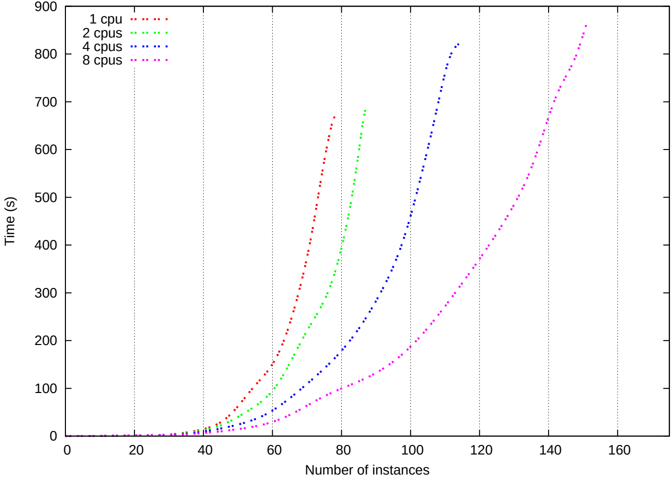
Layout

Instances

- 678 instances taken from SAT Competition 2009
- **170 instances** solvable by STRUCTure
  - Sequential champions go to about 250
- Timelimit: **15 minutes**
  - SAT Competition limit is 20 minutes
  - Enough to understand performance
- Evaluation was done on DAS-4 cluster at VU
  `http://www.cs.vu.nl/das4/`

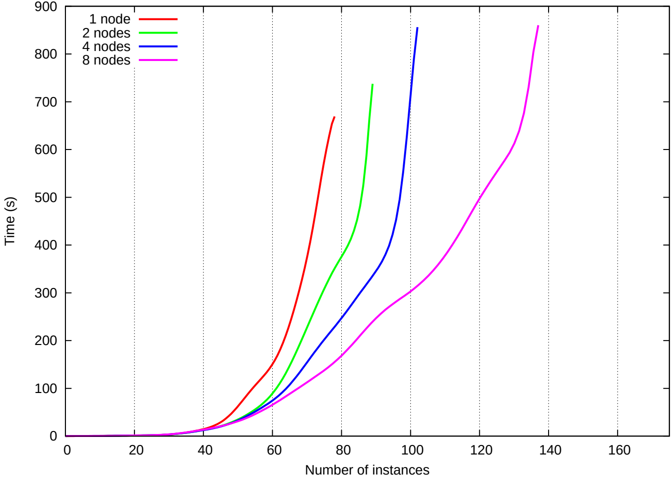
1 to 8 CPUs (1 node)

- using a subset of **175 instances**



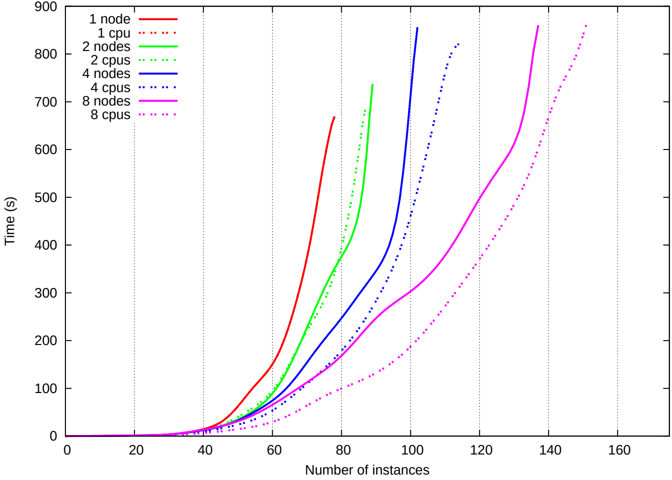
1 to 8 nodes (1 CPU)

- using a subset of **175 instances**



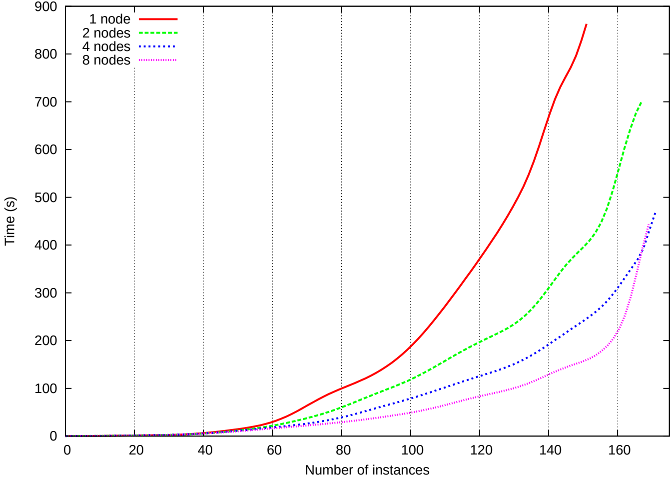
Superimposed

- using a subset of **175 instances**

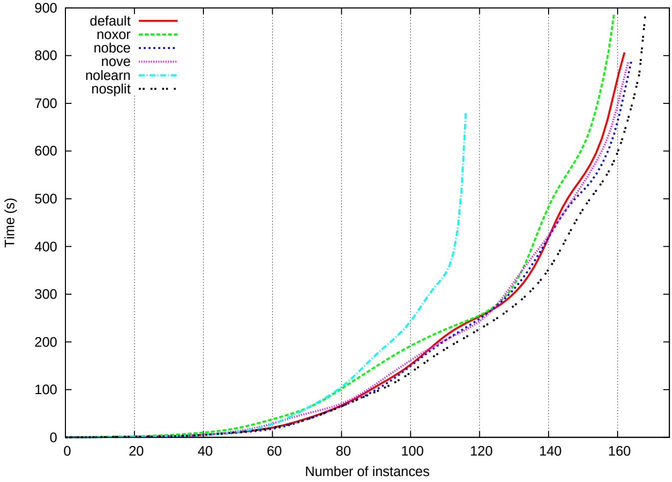


1 to 8 nodes (8 CPUs)

- using a subset of **175 instances**



Disabling various strategies



Selected instances

Instance			Workers (Time:speedup) 1	2	4	8
countbitsrotate016	U	1n	666:1	363:1.8	189:3.5	105:6.3
		1c/n	666:1	397:1.7	247:2.7	191:3.5
AProVE09-19	S	1n	129:1	65:2.0	29:4.4	20:6.5
		1c/n	129:1	117:1.1	70:1.8	62:2.1
instance_n5_i6_pp_ci_ce	U	1n	242:1	94:2.6	59:4.1	36:6.7
		1c/n	242:1	114:2.1	116:2.1	99:2.4
instance_n5_i5_pp	S	1n	102:1	65:1.6	49:2.1	32:3.2
		1c/n	102:1	61:1.7	52:2.0	30:3.4
unif-k3-...	U	1n	1886:1	951:2.0	484:3.9	245:7.7
		1c/n	1886:1	992:1.9	589:3.2	375:5.0
unif-k3-...	S	1n	237:1	91:2.6	43:5.5	13:18.2
		1c/n	237:1	51:4.6	68:3.5	24:9.9

S = Satisfiable, U = Unsatisfiable
1n = 1 node, # cpus varies
1c/n = 1 cpu / node, # nodes varies

Future work

- Find a better variable selection algorithm
- Remove learned clauses
- Reduce the amount of data replication
  - current *overhead* from 10% on big instances, to 40% on small hard instances

Conclusions

- I built a sat solver with horizontal scalability
  - 1 node/1 core performance sacrificed
- with good distributed learning.
- *ManySat* type solvers [11]
  - *don't scale*
  - good performance because of *specialization*
- Constellation makes parallelization very easy
  - As long as your program fits Constellation model, but
  - *No cancellation* can be a problem when you want to stop
  - *Process event and wait* model creates inefficiencies.

[11] ManySat run multiple solvers in parallel

Questions?

Contact

- Can be downloaded from `https://github.com/brtzsnr/structure`
- I can be contacted at `brtzsnr@gmail.com`

XOR Gates and Dependent Variable Removal [12] [13]

- Many instances encode several logical gates: *OR*, *AND*, *XOR*.
- *XOR* with N inputs: $\mathbf{2^N}$ CNF clauses of $N+1$ literals

Idea

Decode XOR gates from formula and use them as eqns. in $\mathbb{Z}_2$.

Dependent Variable Removal

- If a variable appears in exactly *one XOR gate* then the XOR gate *can be removed* from the formula
  - because the variable can always be set to satisfy
- Gaussian Elimination used to get more dependent vars
  - Pick a variable in a gate and knock it out from other gates.

[12] M.J.H. Heule. *Towards a lookahead sat solver for general purposes.*

[13] J.A. Roy, I.L. Markov. *Restoring circuit structure from sat instances.*

Dependendent Variable Removal

Instance [14] [15]	CryptoMinisat 2.6	STRUCTure
...270-2.sat05-2249...	-	0.319
...270-1.sat05-2248...	-	0.306
...250-3.sat05-2220...	-	0.296
...230-2.sat05-2189...	497.877	0.291
...280-1.sat05-2263...	272.806	0.286
...280-3.sat05-2265...	-	0.283
...280-2.sat05-2264...	-	0.281
...210-2.sat05-2159...	50.196	0.246
...220-1.sat05-2173...	44.195	0.231

Table: Times on some instances containing many Dependent Variables

[14] SAT07/crafted/Difficult/contest05/jarvisalo/mod2-rand3bip-sat-....reshuffled-07.cnf

[15] SAT07/crafted/Medium/contest05/jarvisalo/mod2-rand3bip-sat-....reshuffled-07.cnf